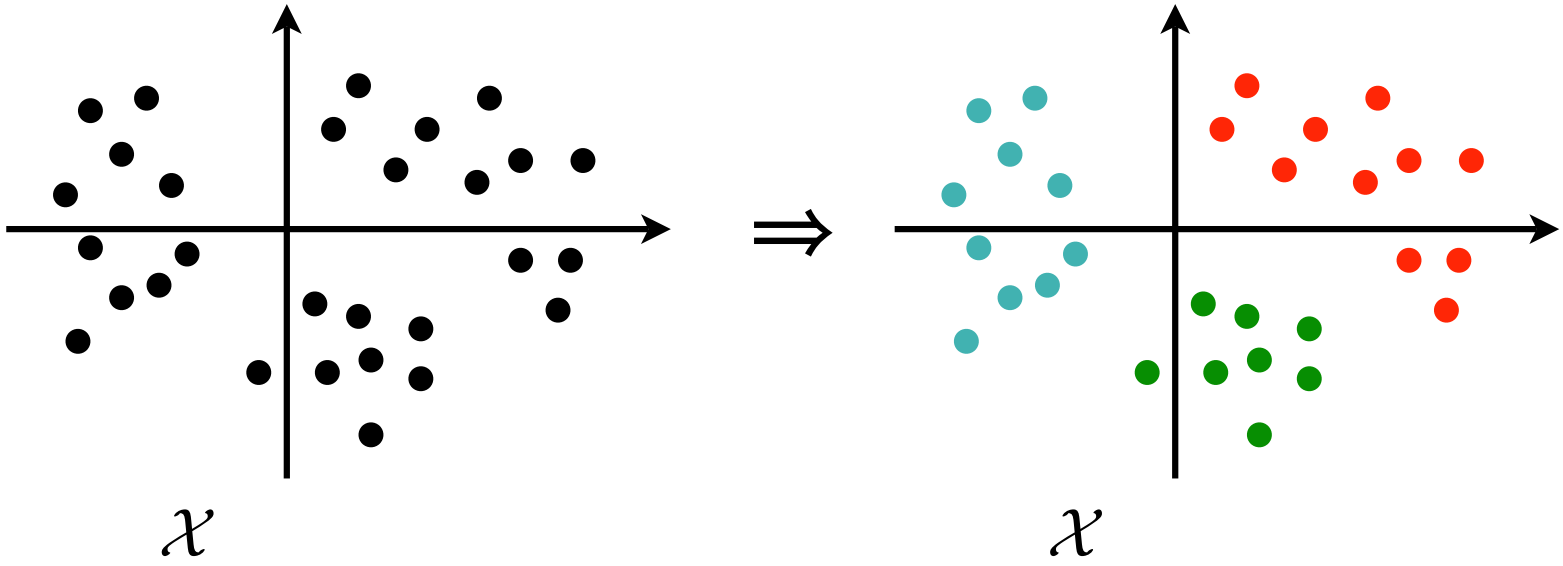
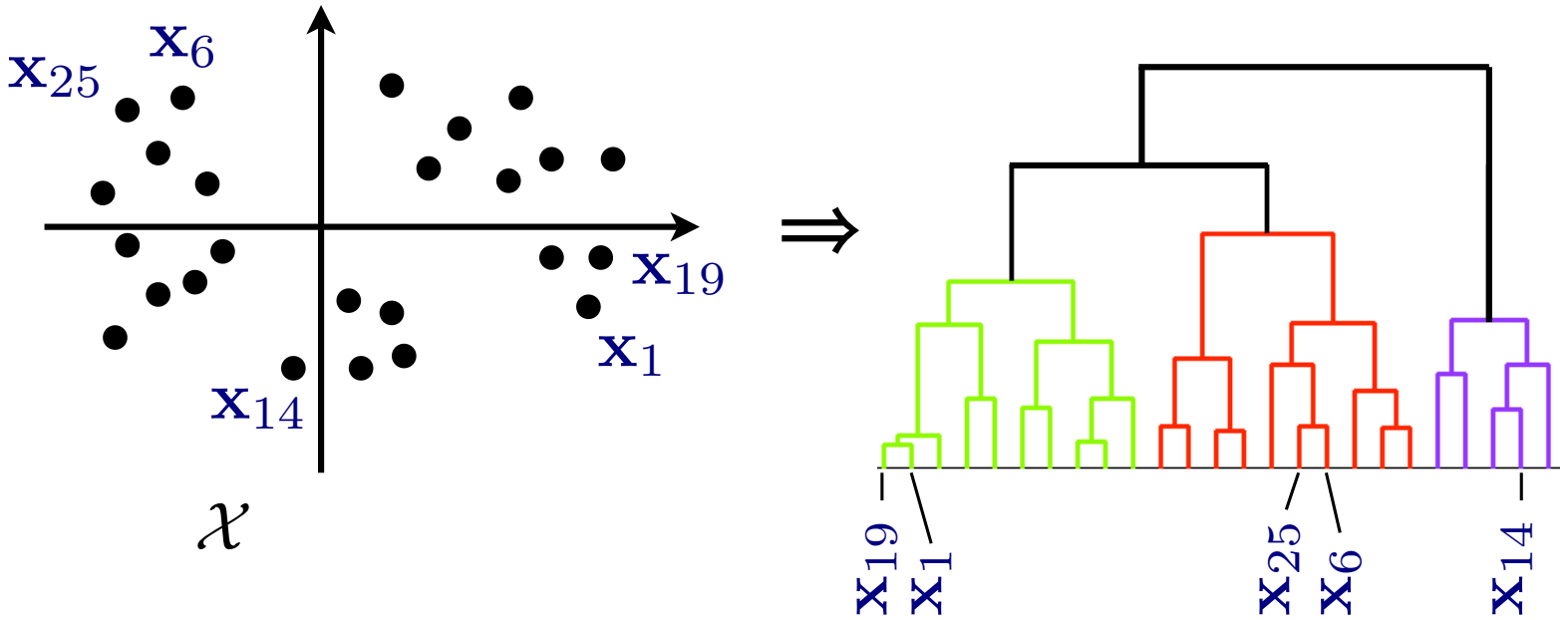
Unsupervised learning: Clustering

Partitional clustering: basic idea

- Each data vector $\mathbf{x}_i$ is assigned to one of K clusters
- Typically K and a proximity measure is selected by the user, while the chosen algorithm then learns the actual partitions
- In the example below, $K = 3$ and the partitions are shown using color (red, green, blue)



Hierarchical clustering: basic idea



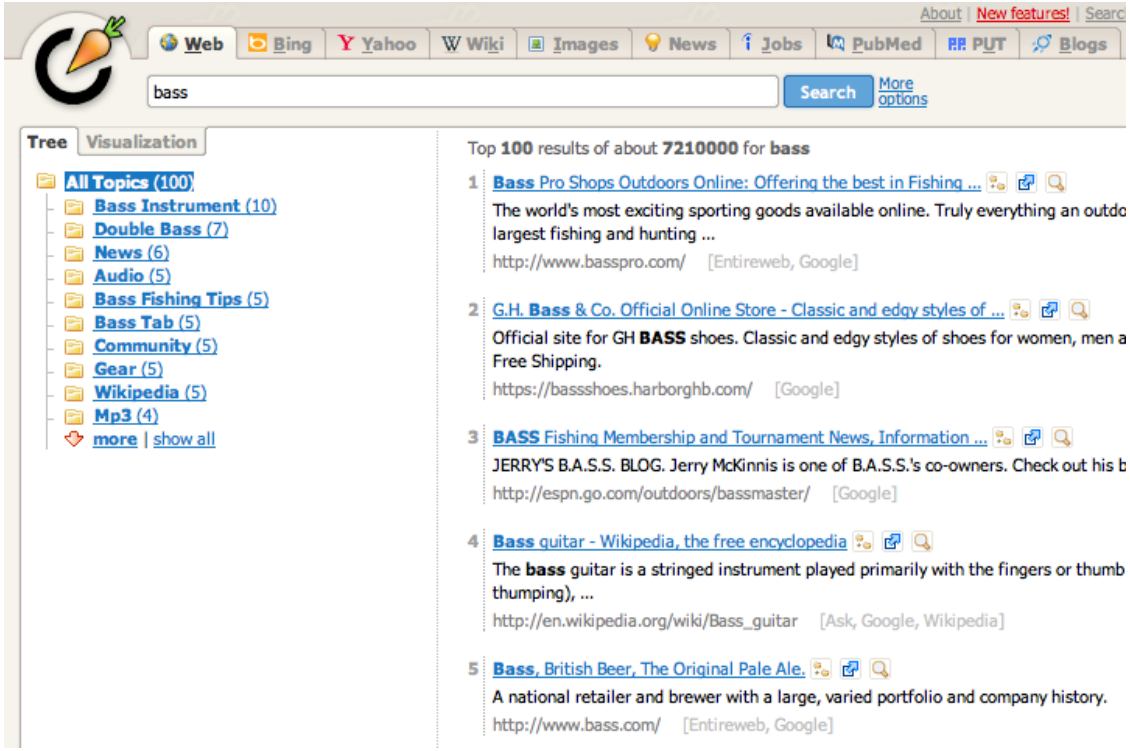
- In this approach, data vectors are arranged in a tree, where nearby ('similar') vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ are placed close to each other in the tree
- Any horizontal cut corresponds to a partitional clustering
- In the example above, the 3 colors have been added manually for emphasis (they are *not* produced by the algorithm)

Motivation for clustering

Understanding the data:

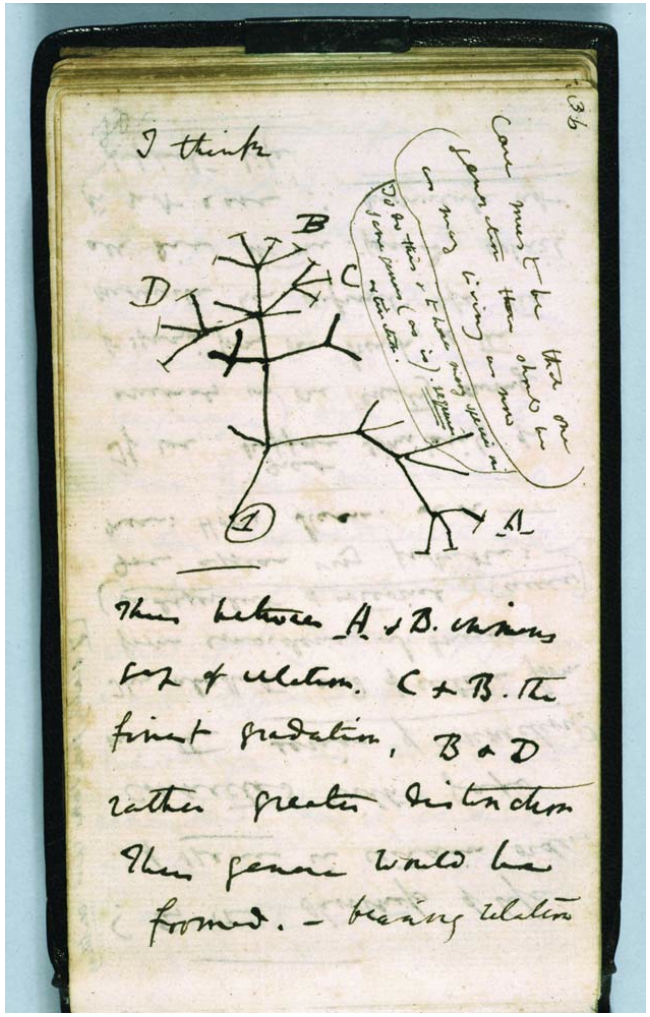
- Information retrieval:

  organizing a set of documents for easy browsing (for example a hierarchical structure to the documents), as we saw in the carrot2 application:



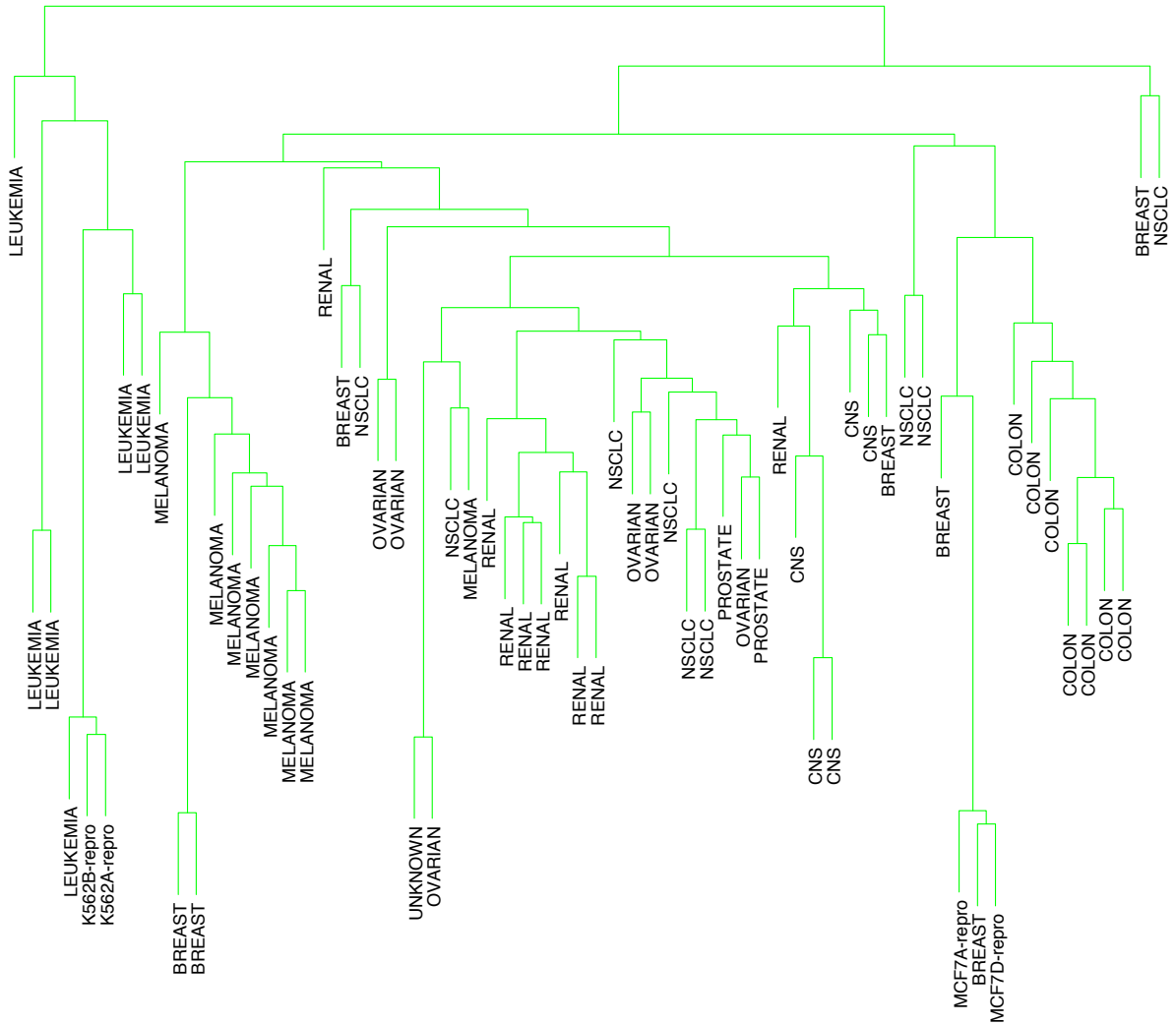
- Biology:

  creating a taxonomy of species, finding groups of genes with similar function, etc



- Medicine:

  understanding the relations among diseases or psychological conditions, to aid in discovering the most useful treatments



- Business:

  grouping customers by their preferences or shopping behavior, for instance for targeted advertisement campaigns

  For example:

  - Customers who follow advertisements carefully, and when in the shop buy only what is on sale
  - Customers who do not seem to react to advertisements at all
  - Customers who are attracted by advertisements, also buy other things in the store while there...

  To whom should you send advertisements?

- Other motivations: simplifying the data for further processing/transmission
  - Summarization:

    reduce the effective amount of data by considering only the prototypes rather than the original data vectors

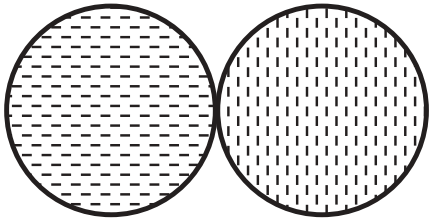
  - 'Lossy' compression:

    saving disk space by only storing a prototype vector which is 'close enough'

What is a cluster?

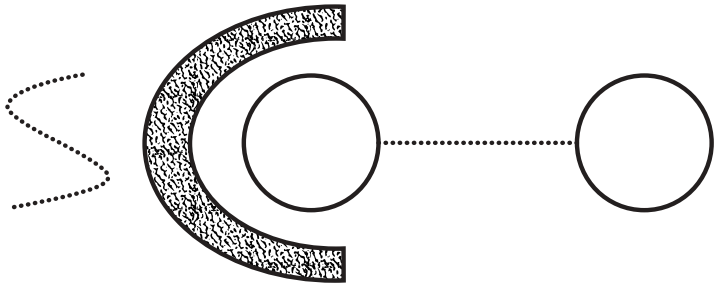
- Clusters are called *well separated* if every point is closer (more similar) to all other points in its cluster than to any point in some other cluster.
- Commonly, clusters are represented by 'cluster prototypes' or 'centers'. In this case it makes sense to require that each point is closer to its cluster prototype than to any other prototype.

(a) Well-separated clusters. Each point is closer to all of the points in its cluster than to any point in another cluster.

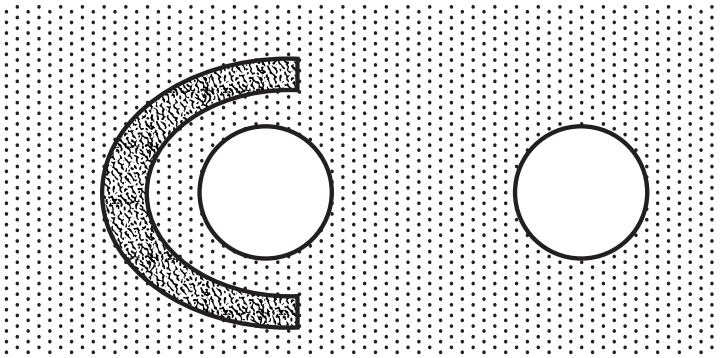


(b) Center-based clusters. Each point is closer to the center of its cluster than to the center of any other cluster.

- We can also define clusters based on contiguity, requiring only that there are no 'gaps' in the clusters.
- Alternatively, we can use the *density* of various regions of the space to define clusters, as in (d) below.

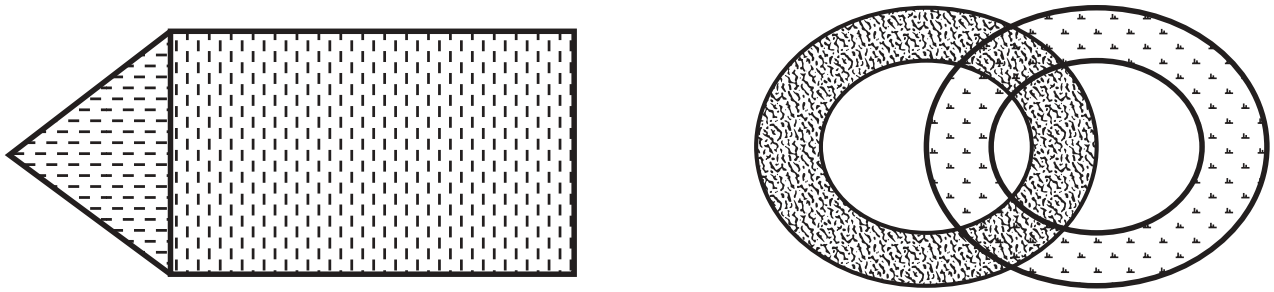


(c) Contiguity-based clusters. Each point is closer to at least one point in its cluster than to any point in another cluster.



(d) Density-based clusters. Clusters are regions of high density separated by regions of low density.

- Finally, we can use more sophisticated (perhaps application-specific) notions of clusters, though in high-dimensional cases such notions may be difficult to identify.



(e) Conceptual clusters. Points in a cluster share some general property that derives from the entire set of points. (Points in the intersection of the circles belong to both.)

K-means

- We now describe a simple and often used partional clustering method: *K-means*
- For simplicity, we will here describe it in the Euclidean space, but extensions are possible (see textbook and exercise set 6)
- Notation:

$\mathbf{x}_i$	the i:th data vector, $i = 1, \ldots, N$
N	the number of data vectors
n	the number of attributes, i.e. length of vector $\mathbf{x}_i$
K	the number of clusters (user-specified)
$\mathbf{c}_j$	the prototype vector for the j:th cluster
a_i	the cluster assignment of data vector $\mathbf{x}_i$. $a_i \in \{1, \ldots, K\}$
C_j	the set of indices i of the $\mathbf{x}_i$ belonging to cluster j, i.e. $C_j = \{i : \ a_i = j\}$

K-means: pseudocode

- Input: A set of N points $\mathbf{x}_i$, and the desired number of clusters K
- Output: A partition of the points into K clusters, i.e. an assignment $a_i \in \{1, \ldots, K\}$ corresponding to each $\mathbf{x}_i$ defining to which cluster each data vector belongs. Also returns the K centroids $\mathbf{c}_j$, $j = 1, \ldots, K$.
- Pseudocode:

Algorithm 8.1 Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

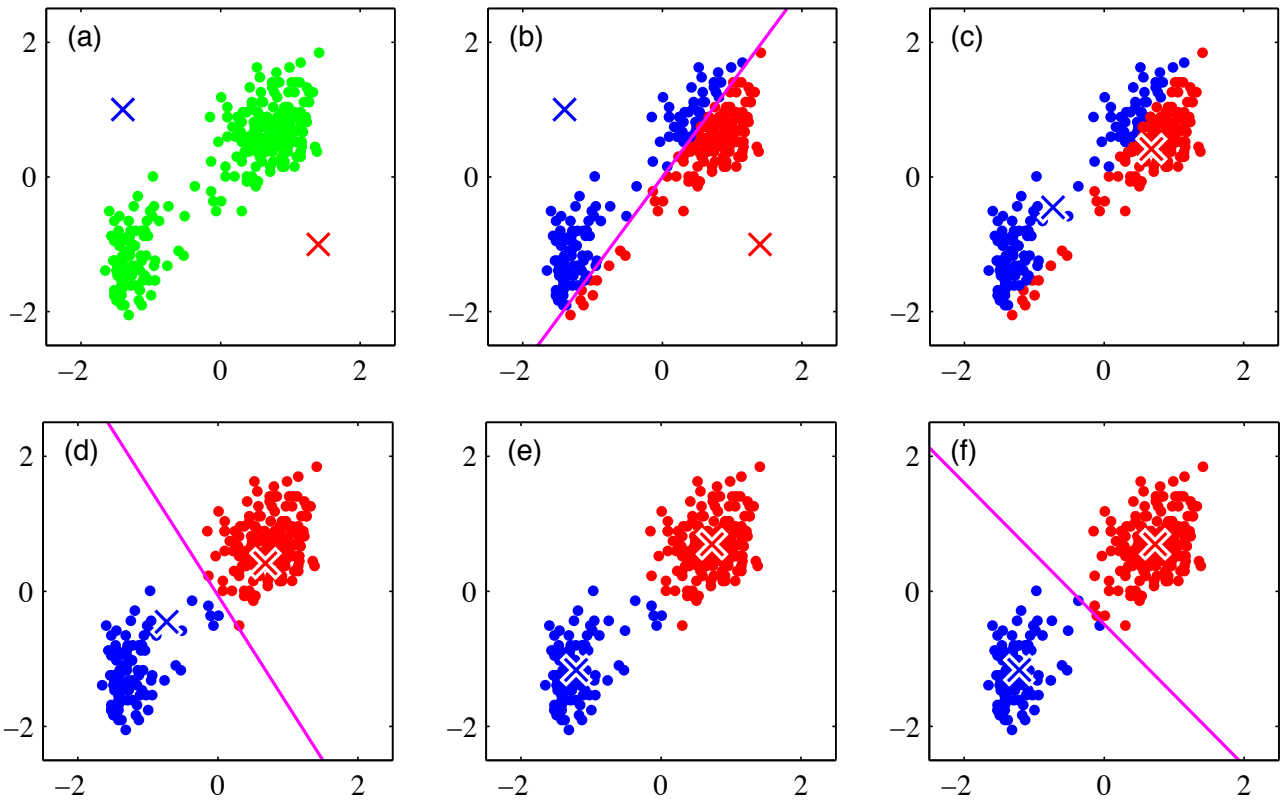
Details:

- In line 1, the simplest solution is to initialize the $\mathbf{c}_j$ to equal K random vectors from the input data
- In line 3, for each datapoint i, set $a_i := \arg\min_j ||\mathbf{x}_i - \mathbf{c}_j||_2$
- In line 4, for each cluster $j = 1, \ldots, K$ we set

$$\mathbf{c}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{x}_i,$$

  i.e. each cluster centroid is set to the mean of the data vectors which were assigned to that cluster in line 3.

K-means: 2D example



- Data from the 'Old faithful' geyser (horizontal axis is duration of eruption, vertical axis is waiting time to the next eruption, both scaled to zero mean and unit variance)

K-means: objective function

- Consider the following measure of the goodness of the clustering

$$\text{SSE} = \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in C_j} ||\mathbf{c}_j - \mathbf{x}_i||_2^2$$

  that is, take the sum of the squared Euclidean distance from each datapoint $\mathbf{x}_i$ to the prototype vector $\mathbf{c}_j$ of the cluster to which it belongs.

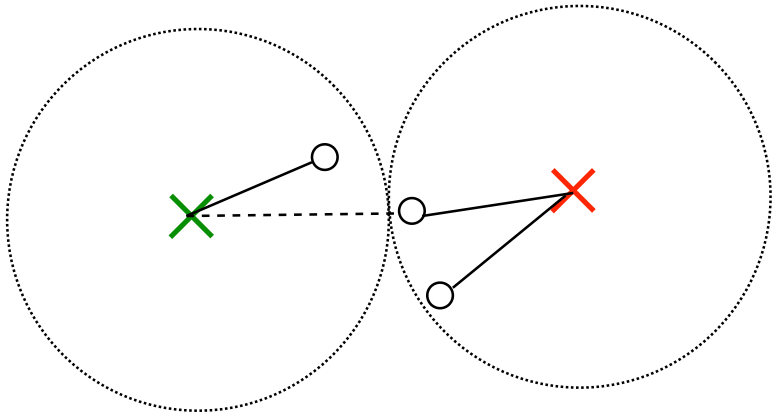
- We will show that in each step of the K-means algorithm the SSE value either stays the same or decreases. (Note: here we aim for ease of understanding rather than give a formal proof with lots of notation.)

- At any point in the algorithm, we have two sets of parameters: The N cluster assignments a_i (which directly determine the C_j), and the K centroids $\mathbf{c}_j$.

- First, we see that, while holding the centroids $\mathbf{c}_j$ fixed, recomputing the assignments a_i such that each datapoint $\mathbf{x}_i$ is assigned to the cluster j wih the closest cluster centroid $\mathbf{c}_j$, i.e.

$$a_i = \arg\min_j ||\mathbf{x}_i - \mathbf{c}_j||_2^2$$

  is the *optimal* clustering of the datapoints in terms of minimizing the SSE, for fixed $\mathbf{c}_j$, $j = 1, \ldots K$.

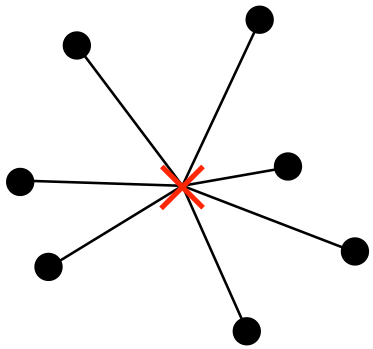


- Hence, regardless of what the cluster assignments were at the *beginning* of step 3 of the algorithm, at the *end* of that step the SSE cannot have increased (as it is now optimal for the given cluster centroids).

- Next, we show a similar property for step 4, namely that for a given cluster assignment, the centroid given by the mean of the data vectors belonging to the cluster

$$\mathbf{c}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i,$$

  is optimal in terms of minimizing the SSE objective.



Isolate a given cluster j. Denote the p:th component of $\mathbf{c}_j$ by c_{jp} and similarly the p:th component of $\mathbf{x}_i$ by x_{ip}. The SSE for this cluster is equal to:

$$\text{SSE}_j = \sum_{\mathbf{x}_i \in C_j} ||\mathbf{c}_j - \mathbf{x}_i||_2^2 = \sum_{\mathbf{x}_i \in C_j} \sum_{p=1}^{n} (c_{jp} - x_{ip})^2$$

Now take the partial derivative of SSE_j with respect to $c_{jp'}$:

$$\begin{aligned}
\frac{\partial \text{SSE}_j}{\partial c_{jp'}} &= \sum_{\mathbf{x}_i \in C_j} \sum_{p=1}^{n} \frac{\partial}{\partial c_{jp'}} (c_{jp} - x_{ip})^2 \\
&= \sum_{\mathbf{x}_i \in C_j} \frac{\partial}{\partial c_{jp'}} (c_{jp'} - x_{ip'})^2 \\
&= \sum_{\mathbf{x}_i \in C_j} 2(c_{jp'} - x_{ip'}) \;=\; 0 \\
&\Rightarrow c_{jp'} = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} x_{ip'},
\end{aligned}$$

- Thus, line 3 select the optimal assignments a_i *given the centroids* $\mathbf{c}_j$, and line 4 selects the optimal centroids $\mathbf{c}_j$ given the assignments a_i (where optimality is with respect to minimizing the SSE)

- Hence, the SSE never increases during the course of the algorithm

- Given that there are a finite number (K^N) of possible assignments, the algorithm is guaranteed to converge to a stable state in a finite number of steps. (In practice, the number of iterations to convergence is typically much smaller than this!)

Space and running time complexity

- Space requirements are modest, as (in addition to the data itself) we only need to store:
  1. The index of the assigned cluster for each datapoint $\mathbf{x}_i$
  2. The cluster centroid for each cluster

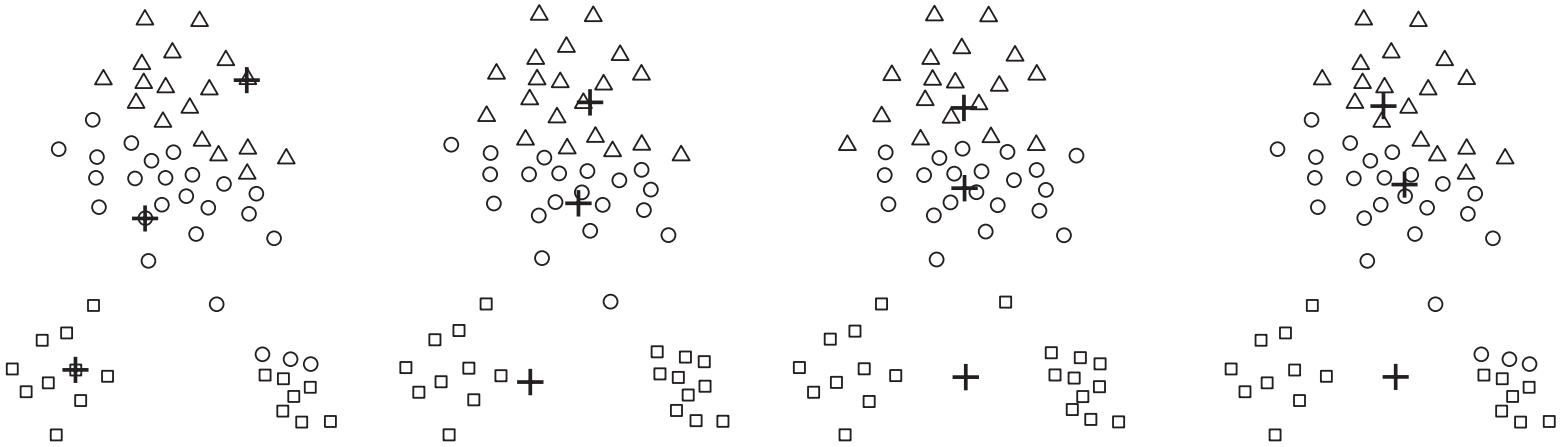
- The running time is linear in all the relevant parameters, i.e. $O(INKn)$, where I is the number of iterations, N the number of samples, K the number of clusters, and n the number of dimensions (attributes).

  (The number of iterations I typically does not depend heavily on the other parameters.)

Influence of initialization

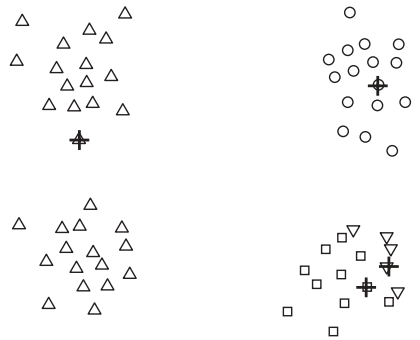
- The algorithm only guarantees that the SSE is non-increasing. It is still local search, and does *not* in general reach the global minimum.

  Example 1:

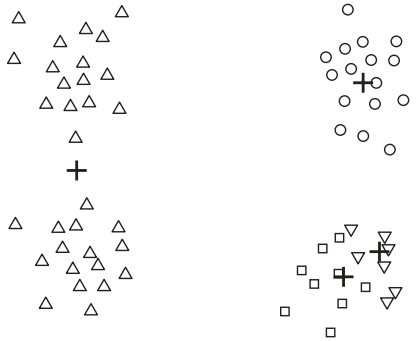


(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

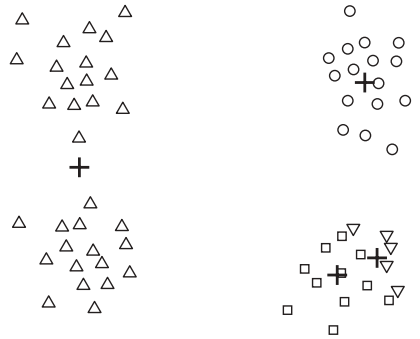
Example 2:



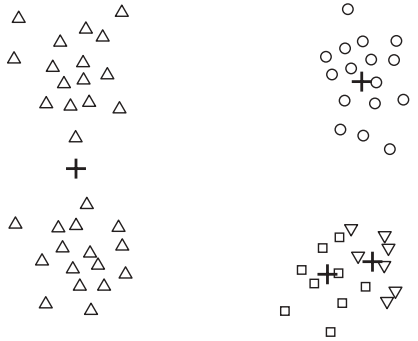
(a) Iteration 1.



(b) Iteration 2.



(c) Iteration 3.



(d) Iteration 4.

- One possible solution: Run the algorithm from many random initial conditions, select the end result with the smallest SSE. (Nevertheless, it may still find very 'bad' solutions almost all the time.)

How to select the number of clusters?

- Not a priori clear what the 'optimal' number of clusters is:

(a) Original points.

(b) Two clusters.

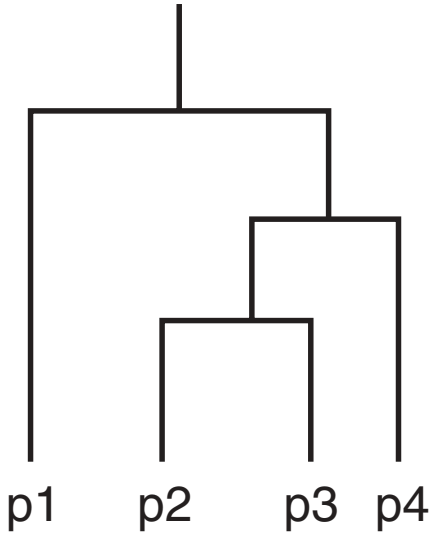
(c) Four clusters.

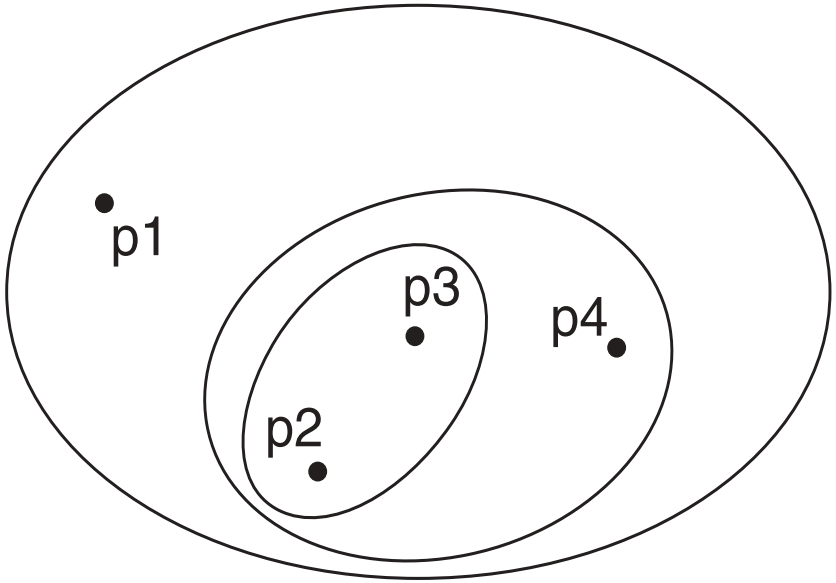
(d) Six clusters.

- The more clusters, the lower SSE, so need some form of 'model selection' approach
- Will discuss this a bit more in the context of clustering validation strategies later

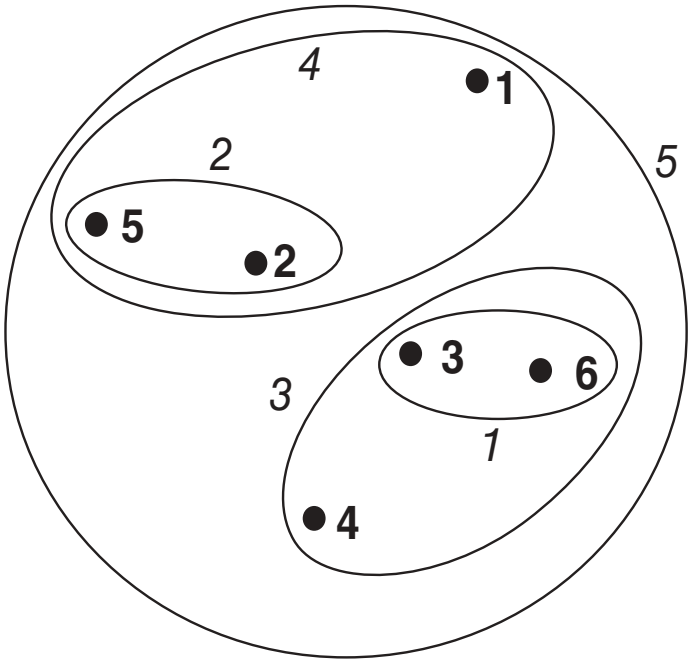
Hierarchical clustering

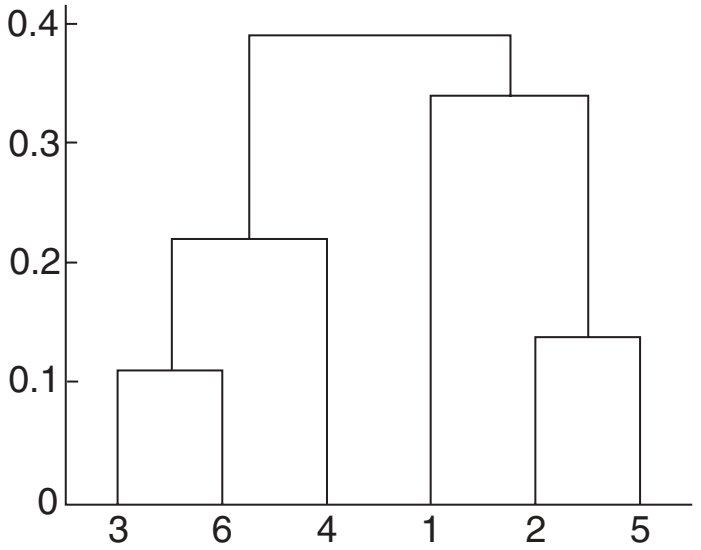
- Dendrogram representation:
  - *Nested* cluster structure
  - Binary tree with datapoints (objects) as leaves
  - Cutting the tree at any height produces a partitional clustering
- Example 1:





- Example 2:





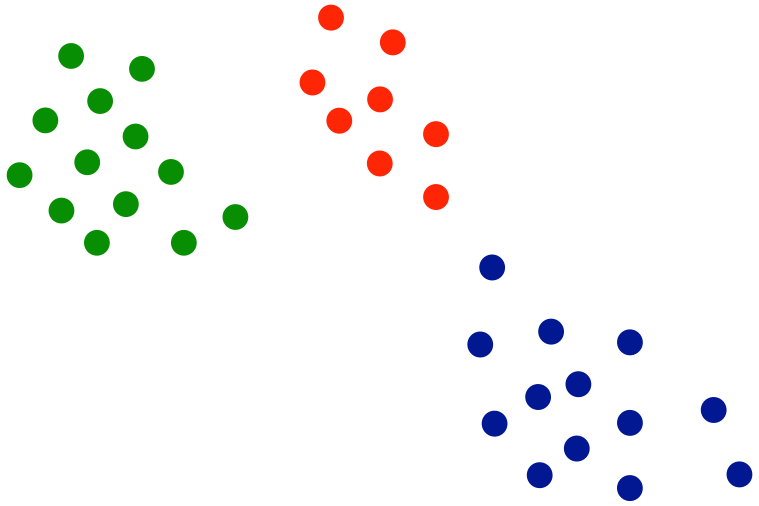
- Height of horizontal connectors indicate the dissimilarity between the combined clusters (details a bit later)

General approaches to hierarchical clustering:

- Divisive approach:
  1. Start with one cluster containing all the datapoints.
  2. Repeat for all non-singleton clusters:
     - Split the cluster in two using some partitional clustering approach (e.g. K-means)
- Agglomerative approach:
  1. Start with each datapoint being its own cluster
  2. Repeat until there is just one cluster left:
     - Select the pair of clusters which are most similar and join them into a single cluster

(The agglomerative approach is much more common, and we will exclusively focus on it in what follows.)

Need a similarity/proximity measure for *pairs of clusters* (in addition to similarity of *pairs of datapoints*). E.g. need to compare $d(C_{\text{red}}, C_{\text{green}})$, $d(C_{\text{red}}, C_{\text{blue}})$, and $d(C_{\text{green}}, C_{\text{blue}})$:



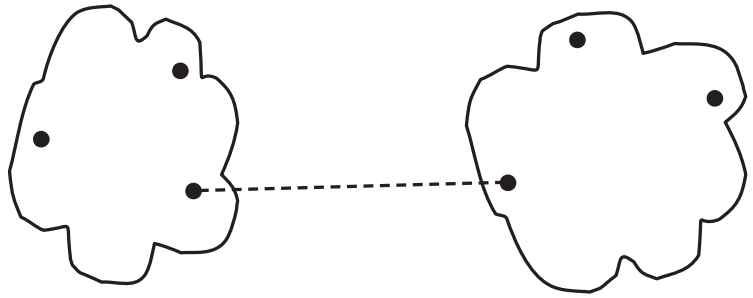
- Notation

$\mathbf{x}_i$	the i:th data vector, $i = 1, \ldots, N$
C_a	the set of indices i of the $\mathbf{x}_i$ belonging to cluster a,
$d(C_a, C_b)$	dissimilarity between clusters C_a and C_b

- 'Single-link' (='MIN')

$$d(C_a, C_b) = \min_{i \in C_a,\, j \in C_b} d(\mathbf{x}_i, \mathbf{x}_j),$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the dissimilarity between the two datapoints (objects) $\mathbf{x}_i$ and $\mathbf{x}_j$.



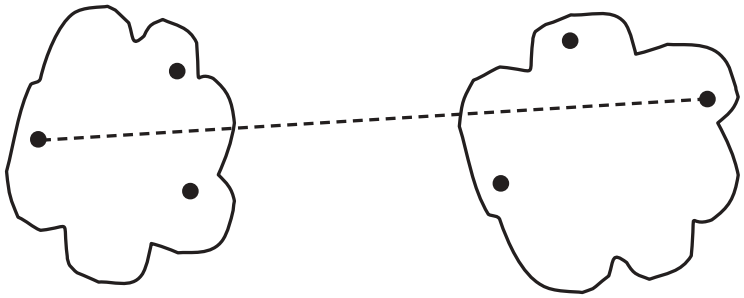
(Note that when working with *similarity* measures $s(\cdot, \cdot)$ we instead take the object pair with *maximum* similarity!)

- Alternatively, we can try enforced that clusters should have *all* pairs of points reasonably close to each other. This gives

  'Complete-link' (='MAX')

$$d(C_a, C_b) = \max_{i \in C_a,\, j \in C_b} d(\mathbf{x}_i, \mathbf{x}_j),$$

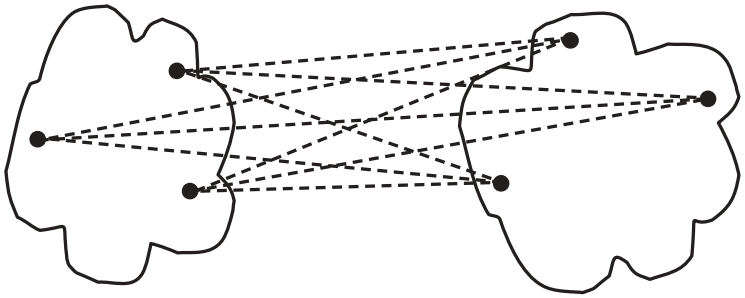
  where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the dissimilarity between the two datapoints (objects) $\mathbf{x}_i$ and $\mathbf{x}_j$.



  (Again, for *similarity* measures $s(\cdot, \cdot)$ we instead take *minimum* of the objectwise similarities!)

- An intermediate criterion is 'Group average'

$$d(C_a, C_b) = \frac{1}{|C_a||C_b|} \sum_{i \in C_a,\, j \in C_b} d(\mathbf{x}_i, \mathbf{x}_j),$$



(With *similarity* measures $s(\cdot, \cdot)$ we also just take the average value.)

- Centroid-based hierarchical clustering:

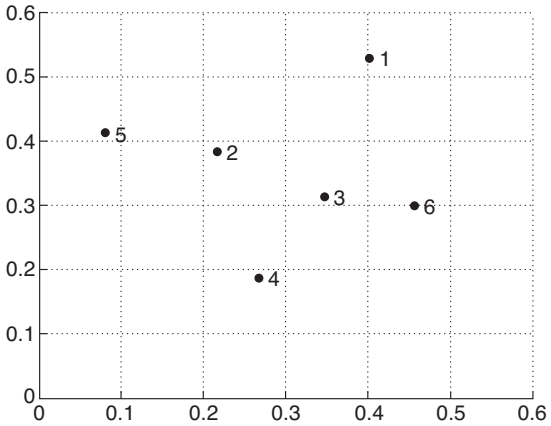
$$d(C_a, C_b) = d(\mathbf{c}_a, \mathbf{c}_b),$$

where the prototypes $\mathbf{c}_a$ and $\mathbf{c}_b$ are the cluster prototypes given by the means of the vectors in each cluster:

$$\mathbf{c}_a = \frac{1}{|C_a|} \sum_{i \in C_a} \mathbf{x}_i \quad \text{and} \quad \mathbf{c}_b = \frac{1}{|C_b|} \sum_{i \in C_b} \mathbf{x}_i$$

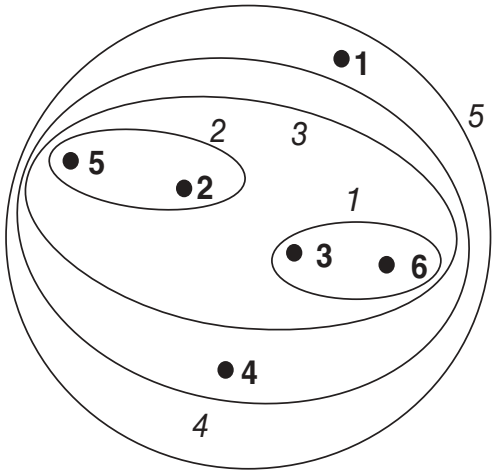
- *Ward's method* is based on using prototypes (centroids) for each cluster, and measuring the dissimilarity between clusters as the increase in SSE (sum of squared errors from datapoints to their prototype) resulting from combining the two clusters

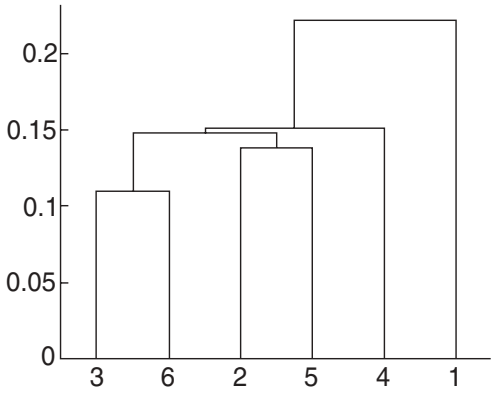
Example 1:



	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

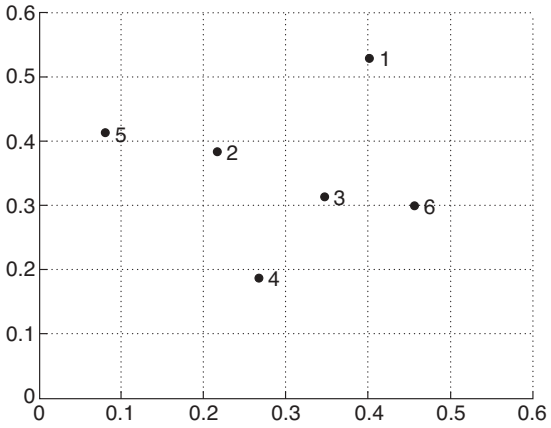
- Single-link:





(The heights in the dendrogram correspond to the dissimilarities $d(C_a, C_b)$ when clusters C_a and C_b are combined.)

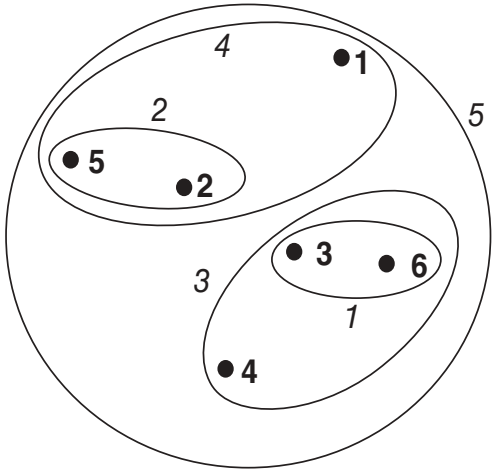
Example 2:

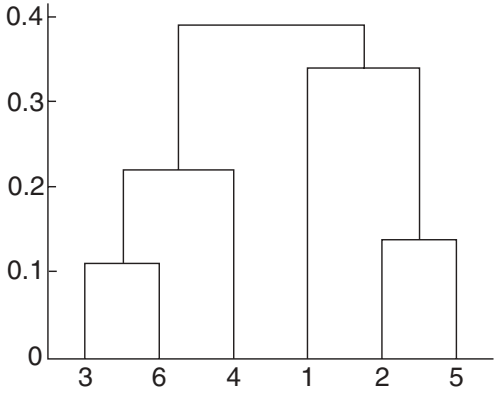


	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

- Complete-link:

  (The heights in the dendrogram correspond to the dissimilarities $d(C_a, C_b)$ when clusters C_a and C_b are combined.)





- Cluster shapes:
  - *Single-link* can produce arbitrarily shaped clusters (joining quite different objects which have some intermediate links that connect them)
  - *Complete-link* tends to produce fairly compact, globular clusters. Problems with clusters of different sizes.
  - *Group average* is a compromise between the two

single link complete link

- Lack of a global objective function:
  - In contrast to methods such as K-means, the agglomerative hierarchical clustering methods do not have a natural objective function that is being optimized. Even Ward's method does not give even local minima in terms of minimizing the SSE!

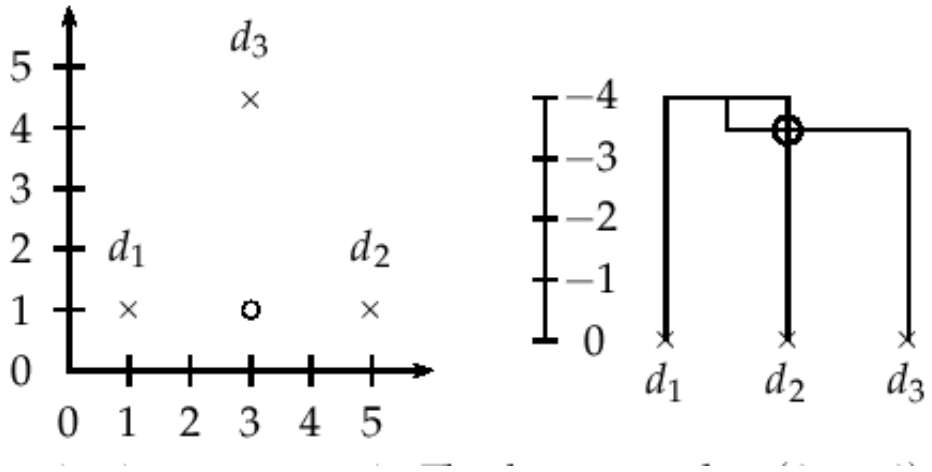
- *Monotonicity:*

  If the dissimilarity between a pair clusters merged at any point in the algorithm is always at least as large as the dissimilarity of the pair of clusters merged in the previous step, the clustering is *monotonic*.

  - Single-link, complete-link, and group average: Yes!
  - Centroid-based hierarchical clustering: No! Example:

    $d_1 = (1+\epsilon, 1)$, $d_2 = (5, 1)$, $d_3 = (3, 1+2\sqrt{3})$.
    The first combination (of d_1 and d_2) occurs at a distance of $4-\epsilon$. The point $o = (3+\epsilon/2, 1)$.
    The next combination occurs at distance of

    $$\sqrt{(2\sqrt{3})^2 + (\epsilon/2)^2} \approx 2\sqrt{3} \approx 3.4641 < 4-\epsilon$$



- Computational complexity
  - The main storage requirement is the matrix of pairwise proximities, containing a total of $N(N-1)/2$ entries for N datapoints. So the space complexity is: $O(N^2)$.
  - Computing the proximity matrix takes $O(N^2)$. Next, there are $O(N)$ iterations, where in each one we need to find the minimum of the pairwise dissimilarities between the clusters. Trivially implemented this would lead to an $O(N^3)$ algorithm, but techniques exist to avoid exhaustive search at each step, yielding complexities in the range $O(N^2)$ to $O(N^2 \log N)$.

    (Compare this to K-means, which only requires $O(NK)$ for K clusters.)

Hence, hierarchical clustering is *directly* applicable only to relatively small datasets.